Московский государственный технический университет им. Н.Э. Баумана

Кафедра «Системы обработки информации и управления»

Рубежный контроль №2

по дисциплине

«Методы машинного обучения»

Выполнил:

студент группы ИУ5И-21М

Ван Чжэн

Москва — 2024 г.

Я выбрал набор данных-«продажа подержанных автомобилей».

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import ComplementNB
```

```
[4] df = pd.read_csv('/content/second_hand_car_sales.csv')
df
```

	Manufacturer	Model	Engine Size (L)	Fuel Type	Year of Manufacture	Mileage	Price
0	Mercedes-Benz	Cruze	1.418475	Electric	2013	61837	34792
1	Toyota	A4	4.492330	Electric	2003	128993	27129
2	Audi	C-Class	4.739375	Electric	2000	81362	29141
3	Nissan	Model 3	3.128423	Petrol	2011	168204	24731
4	Mercedes-Benz	Golf	1.650279	Diesel	2006	119405	27493
...	...	...	...	...	...	...	...
49995	Chevrolet	Corolla	1.241130	Diesel	2021	163295	1110
49996	Nissan	Civic	3.741902	Electric	2012	85805	27877
49997	Toyota	Altima	2.501539	Hybrid	2016	187733	42132
49998	Ford	Model 3	2.066934	Electric	2022	136728	39121
49999	Chevrolet	Fiesta	3.261779	Petrol	2000	104033	42007

50000 rows × 7 columns

Классификатор№1: KNeighborsClassifier

KNeighborsClassifier+Tfidf

KNeighborsClassifier+TFIDF

```
from sklearn.neighbors import KNeighborsClassifier

X = df['Manufacturer']
y = df['Fuel Type']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 使用 TfidfVectorizer 进行向量化
tfidf_vectorizer = TfidfVectorizer()
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)

# 使用 KNeighborsClassifier 进行分类（基于 TfidfVectorizer）
knn_tfidf = KNeighborsClassifier()
knn_tfidf.fit(X_train_tfidf, y_train)
y_pred_tfidf = knn_tfidf.predict(X_test_tfidf)
print(classification_report(y_test, y_pred_tfidf, digits=4))
```

```
              precision    recall  f1-score   support

      Diesel     0.2403    0.3918    0.2979      2478
    Electric     0.2494    0.4854    0.3295      2530
      Hybrid     0.0000    0.0000    0.0000      2547
      Petrol     0.2363    0.1002    0.1407      2445

    accuracy                         0.2444     10000
   macro avg     0.1815    0.2444    0.1920     10000
weighted avg     0.1804    0.2444    0.1916     10000
```

KNeighborsClassifier+CountVec

KNeighborsClassifier+CountVec

```
# 使用 CountVectorizer 进行向量化
count_vectorizer = CountVectorizer()
X_train_count = count_vectorizer.fit_transform(X_train)
X_test_count = count_vectorizer.transform(X_test)

knn_count = KNeighborsClassifier()
knn_count.fit(X_train_count, y_train)
y_pred_count = knn_count.predict(X_test_count)
print(classification_report(y_test, y_pred_count, digits=4))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: U
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

      Diesel     0.2360    0.3793    0.2910      2478
    Electric     0.2540    0.4107    0.3138      2530
      Hybrid     0.2425    0.1834    0.2088      2547
      Petrol     0.0000    0.0000    0.0000      2445

    accuracy                         0.2446     10000
   macro avg     0.1831    0.2433    0.2034     10000
weighted avg     0.1845    0.2446    0.2047     10000
```

Классификатор№2: LogisticRegression

LogisticRegression+TFIDF

```
from sklearn.linear_model import LogisticRegression

# 使用 LogisticRegression 进行分类（基于 TfidfVectorizer）
lr_tfidf = LogisticRegression()
lr_tfidf.fit(X_train_tfidf, y_train)
y_pred_lr_tfidf = lr_tfidf.predict(X_test_tfidf)
print(classification_report(y_test, y_pred_lr_tfidf, digits=4))
```

```
              precision    recall  f1-score   support

      Diesel     0.2497    0.3067    0.2753      2478
    Electric     0.2460    0.0976    0.1398      2530
      Hybrid     0.2419    0.0876    0.1286      2547
      Petrol     0.2507    0.5157    0.3374      2445

    accuracy                         0.2491     10000
   macro avg     0.2471    0.2519    0.2203     10000
weighted avg     0.2470    0.2491    0.2188     10000
```

LogisticRegression+CountVec

LogisticRegression+CountVec

```
# 使用 LogisticRegression 进行分类（基于 CountVectorizer）
lr_count = LogisticRegression()
lr_count.fit(X_train_count, y_train)
y_pred_lr_count = lr_count.predict(X_test_count)
print("\nLogisticRegression 基于 CountVectorizer 的分类报告：")
print(classification_report(y_test, y_pred_lr_count, digits=4))
```

```
LogisticRegression 基于 CountVectorizer 的分类报告：
              precision    recall  f1-score   support

      Diesel     0.2497    0.3067    0.2753      2478
    Electric     0.2460    0.0976    0.1398      2530
      Hybrid     0.2419    0.0876    0.1286      2547
      Petrol     0.2507    0.5157    0.3374      2445

    accuracy                         0.2491     10000
   macro avg     0.2471    0.2519    0.2203     10000
weighted avg     0.2470    0.2491    0.2188     10000
```